Proceedings of the 1993 Winter Simulation Conference
G.W. Evans, M. Mollaghasemi, E.C. Russell, W.E. Biles (eds.)

AN INTERACTIVE GRAPHICAL MODELING TOOL FOR PERFORMANCE AND PROCESS SIMULATION

Dennis S. Mok
Bellcore
Piscataway, NJ 08854

Cynthia A. Funka-Lea
AT&T Bell Laboratories
Holmdel, NJ 07733

ABSTRACT

This paper describes an interactive visual modeling and simulation environment, Q+^{TM} (Q+ is a trademark of AT&T Bell Laboratories) and its COMPASS modeling interface, for performance analysis and simulation of distributed computer systems. In the first section, the authors give an expository description of the Q+ simulation tool. After that Q+'s modeling capability is demonstrated through the use of a distributed order processing system modeling example. The authors complete the paper by presenting the current advances on developing the COMPASS interface to simplify distributed systems modeling by system planners and designers.

1 INTRODUCTION

With the recent advances in personal computer, workstation and high speed network capabilities, the basic architecture for a computer system is now typically consisted of a high performance host system working in concert with client-server type systems. These complex distributed systems consist of multiple high power workstations acting as servers and connected via local area networks (LANs) to groups of graphical personal computers or terminals. To assure that these complex distributed computer systems are optimally configured, and the business operations supported by the systems are properly designed, we need to analyze (through modeling and simulating) the performance of the distributed systems to help make critical configuration and design decisions. One of the means is by visual simulation as described in this paper. Visual modeling tools can help system builders assure the performance of these complex software systems. Visual models let them represent and study the system's operational behavior. The idea of interactive modeling and simulation is that a designer input a model just as he/she would describe it to a colleague: by drawing a picture. Then the system's behavior can be observed via the animated movement of operational entities within the model and the gradual evolution of statistics. In this paper, the design and functionalities of Q+ is described and its modeling capability demonstrated using a distributed systems modeling example. At the end of this paper, current advances on developing a modeling interface (COMPASS) for convenient modeling of distributed computer systems are described.

2 INTERACTIVE VISUAL MODELING AND SIMULATION IN Q+

Q+ is a descendent of the Performance Analysis Workstation (PAW), also developed at AT&T Bell Laboratories by Melamed and Morris (1985). PAW differed from other visual modeling tools in several important respects. It let you completely specify a complex model using graphical operations, execute it immediately without any compilation delay, and observe model animation in the same window in which you built it without any additional effort. You could stop the model in mid-run, change it structurally and parametrically, and then continue the simulation run without delay.

Q+, among other things, supports two basic principles: interactive model building and exploratory analysis of model behavior. Detailed descriptions of Q+ can be found in the Q+ User's Guide and Reference Manual and in an article by Funka-Lea, Kontogiorgos, Morris and Rubin (1991). Q+ has both Monte Carlo simulation capabilities and interfaces to analytic tools. The user interface is driven by popup menus. Q+ has an object oriented design where icons on the screen represent concrete objects (nodes, transactions, statistics, and so on). Q+ has six basic components: graphics editor, text editor, Monte Carlo simulator, the language interpreter, subnetworks, and the utility set. These components are described in the following sections.

2.1 The Graphics Editor

A new model is most easily created using the graphics editor. Using the mouse, you draw the individual network nodes, then connect them to specify the topology (Figure 1). Node drawing is simple, because all icons are preprogrammed and you simply change their position, size and orientation with the mouse. Q+ generates a distinct label for each node, and every node can have a source and a sink. Experimental versions of Q+ allow you to define your own icons for nodes and other objects. To reserve screen space for statistical displays, you sweep out rectangular windows. Q+ allows you to bind a statistic to a window, to clear a window of a Q+ statistics are in three formats: summaries, time-series and histograms. The views of statistics that you select for graphical display are updated every time the screen is refreshed, or when they change, whichever comes first.

2.2 The Text Editor

You can use the Q+ text editor to parameterize a model, define its statistics, and enter expressions. You enter text data in captioned fields of appropriate forms. The editor checks every lexical token thoroughly for syntactic and semantic errors and consistency violations. For example, it checks node and transaction labels for uniqueness, routing for discrepancies with the drawn topology, and discrete distributions for violation of the limit of 1 in the sum of probabilities. The editor flags an error at the first erroneous character, sounding a double beep and rejecting the character, or using an appropriate default for that entry.

2.3 The Monte Carlo Simulator

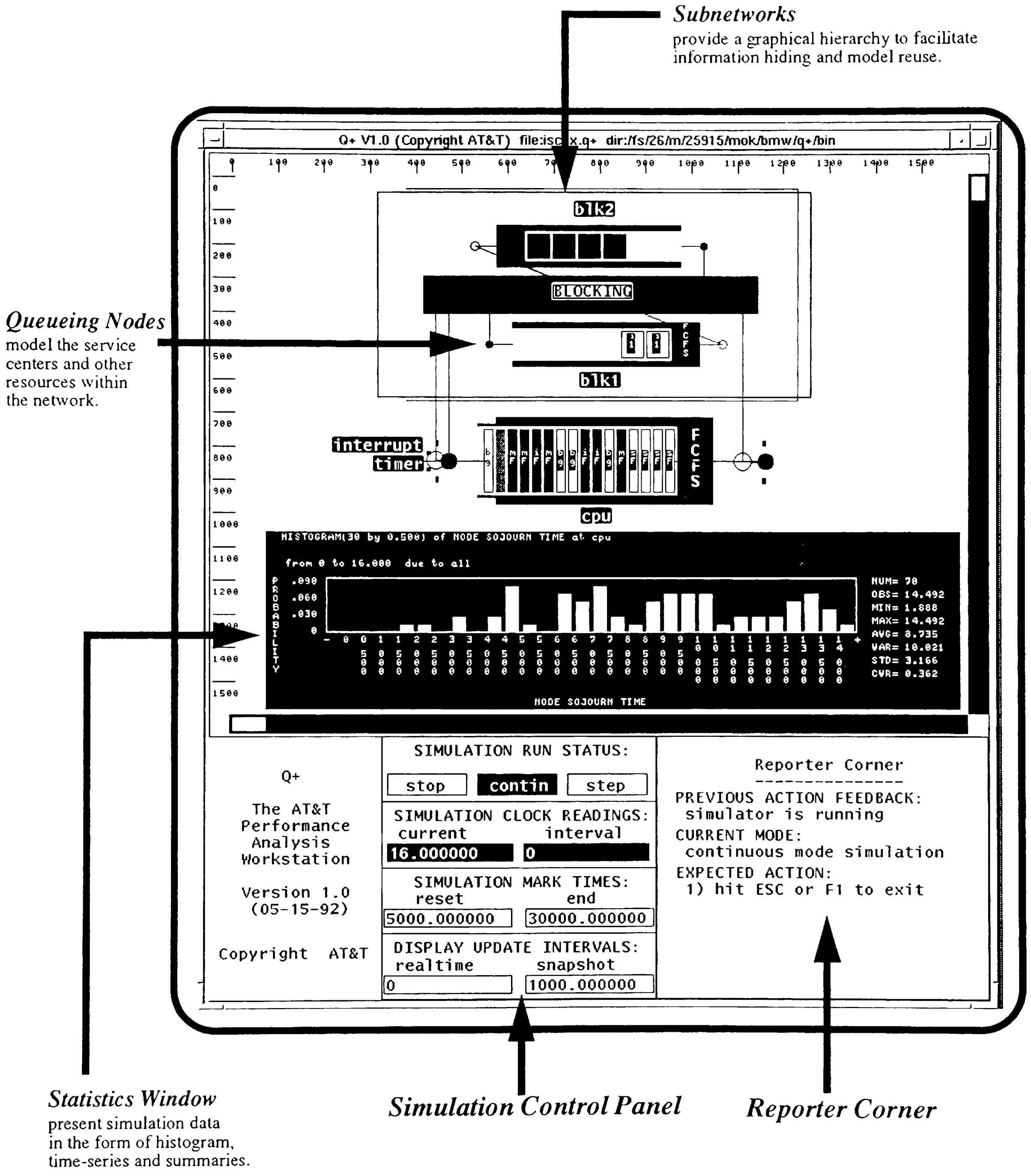
Q+ supports run modes and options to suit the requirements of various stages in the modeling process. Simulation activities are largely controlled from a simulation control panel, where you can set the snapshot window to any nonnegative value. To examine the simulation event by event, you set the snapshot window to zero. Then screen animation displays the shuttling of transactions among nodes, their creation and destruction, and more complicated behavior such as splitting, joining and yanking.

You can step through events or run the animation continuously. When you set the snapshot window to a positive value, the screen is refreshed every snapshot interval to give a time-lapse account of model behavior at the specified time granularity. To launch production runs in batch mode, you select the appropriate option and specify an output file. You can interrupt and restart at any time to track the progress of a batch simulation.

In a more complicated situation, the user may wish to run replications, step through a model's parameters for a sensitivity analysis, or compare multiple models. To facilitate these kinds of activities, Q+ provides a C language library called the C Programming Interface or CPI (formerly called HPE) that allows a model's parameters or structure to be accessed or changed from a C program.

2.4 The Language Interpreter

Although Q+ is a visual modeling tool, it does allow transactions to have textual (data and program) attributes. You can enter most of the static attributes of a transaction, node or model as expressions, and those expressions can access and affect other model entities. These expressions are evaluated by an interpreter called Exp. For example, because performance models are quantitative, it might make sense for a transaction to carry a variable x, which could represent size, history, or system state, and for the network to alter its treatment of this transaction on the basis of the variable. Or this attribute might instead belong to the node and even be a function that the node executes to determine how to route or serve the transaction. The overall effect is a natural complement to the graphical paradigm that can easily express complex operations or protocols.


Q+ V1.0 (Copyright AT&T) file:iscx.q+ dir:/fs/26/m/25915/mok/bmw/q+/bin
blk2
BLOCKING
FCFS
blk1
interrupt
timer
FCFS
cpu
HISTOGRAM(30 by 0.500) of NODE SOJOURN TIME at cpu
from 0 to 16.000 due to all
PROBABILITY
NODE SOJOURN TIME

```
NUM= 78
OBS= 14.492
MIN= 1.888
MAX= 14.492
AVG= 8.735
VAR= 10.021
STD= 3.166
CVR= 0.362
```

```
Q+

The AT&T
Performance
Analysis
Workstation

Version 1.0
(05-15-92)

Copyright  AT&T
```

```
SIMULATION RUN STATUS:
 stop   contin   step
SIMULATION CLOCK READINGS:
current          interval
16.000000        0
SIMULATION MARK TIMES:
reset            end
5000.000000      30000.000000
DISPLAY UPDATE INTERVALS:
realtime         snapshot
0                1000.000000
```

```
Reporter Corner
---------------
PREVIOUS ACTION FEEDBACK:
 simulator is running
CURRENT MODE:
 continuous mode simulation
EXPECTED ACTION:
 1) hit ESC or F1 to exit
```

Subnetworks
provide a graphical hierarchy to facilitate information hiding and model reuse.

Queueing Nodes
model the service centers and other resources within the network.

Statistics Window
present simulation data in the form of histogram, time-series and summaries.

Simulation Control Panel

Reporter Corner

Figure 1. The Q+ Simulation Display

2.5 The Modeling Subnetworks

Many users build large models containing repetitive components, and others like to build their models in a hierarchical fashion, using a top/down or bottom/up approach. The subnetwork capability lets you build and structure models hierarchically. Thus, any model can possess submodels to any degree of nesting. This abstraction facilitates model construction by the well-accepted structured-programming principle of information hiding. It also supports such basic operations as of replicating model components, altering the graphical representation of model parts, and reading in models from libraries. Because Q+ structures models both modularly and hierarchically, this encourages sharing and reuse of modeling efforts.

Q+ subnetworks let you move and rescale subnetworks, and also cut and paste together Q+ models. You can align time and disambiguate names when models are pasted together. You can also manipulate much larger models easily by an enhanced viewing mechanism. The theoretical node limit imposed by graphics exceeds one million, however, memory constraints usually impose a smaller limit.

2.6 The Q+ Utilities

Q+ provides an array of utilities, mostly for file manipulation, translation and simulation. You can save Q+ models from or read them to the graphics screen. You can display a textual listing of an entire Q+ model or save a screen image in a file and print it as a hard copy. Q+ has interfaces to two analytical packages: Panacea (developed by Ramakrishnan and Mitra (1982)), which includes an asymptotic expansion method to solve large steady state queueing networks and QNA (developed by Whitt (1983)), which generates an approximate analytical solution for steady state queueing networks based on the first two moments of each modeling distribution.

2.7 The Q+ World View

Q+'s most primitive world view, that of a queueing network, is an extension of BCMP by Baskett, Chandy, Muntz and Palacios (1975) and Kelly Networks by Kelly (1979). The Q+ simulator is a discrete-event simulation system operating in a world that consists mainly of two fundamental entities: nodes (components of a computer system, geographical sites in a network, and so on) and transactions (jobs, customers, and so on) which circulate among nodes.

A node has associated with it a queue whose positions can hold transactions. A queue's capacity can be positive and finite, or infinite. Nodes have incoming and outgoing paths, so that a queueing network functions like a directed graph. A special kind of node, called an environment node represents sources and sinks. While nodes are static entities usually anchored to network locations, transactions are dynamic entities that circulate and move from node to node. Each transaction has two fundamental attributes associated with it: class tag and family membership. A transaction class tag is a dynamic attribute that is assigned to the transaction whenever it enters a regular node. Typically, class tags are used to denote the state of a transaction during the course of its life in the network. You can specify node and class-dependent priority for transactions. A transaction's class at a given node determines (possibly via an expression) the external arrivals (from a source), service delays and routing decisions for that transaction. A transaction may change class on routing to another node.

Family membership is a static attribute. The family concept helps when a transaction spawns a batch and the batch members must be eventually joined to recover the original transaction (for example, messages which are split into interleaving packets that must be reassembled at their destination). Family membership can propagate and expand by further splitting of transactions. In Q+, there are no special types of transactions. For example, if you want a transaction to represent a token, you simply set its service time to infinity with probability one. Specialized modeling constructs are embodied in the various types of nodes supported by Q+. These are first-come-first-served, last-come-last-served, infinite server (usually representing pure delay), several preemptive-resume disciplines, split nodes (where a transaction may be divided into multiple copies), join nodes (where transactions are merged into a single one) and yank nodes (allowing transactions to pull other transactions from various nodes and reroute them to other nodes, possibly in batches). You can specify split, join, and yank operations by class and family.

3 AN ORDER PROCESSING SYSTEM MODELING EXAMPLE

In this example, we describe an order processing system application modelled in Q+. This application typifies a generic tele-marketing system where the products for sale could be anything from clothing to computers. The model in Figure 2 shows a marketing representative interacting with customers on the phone to take orders. After the order form has been filled electronically, the order is sent via a Local Area Network (LAN) to an order manager who's responsibilities are:

1. Printing a hard copy and filing of the order.

2. Accessing a computerized inventory database system to check on availability of components needed to assemble the order.

3. Assigning the manual work force to process the order if needed.

After these steps are completed, the order is electronically sent to the billing department's database system so the customer is promptly and correctly billed.

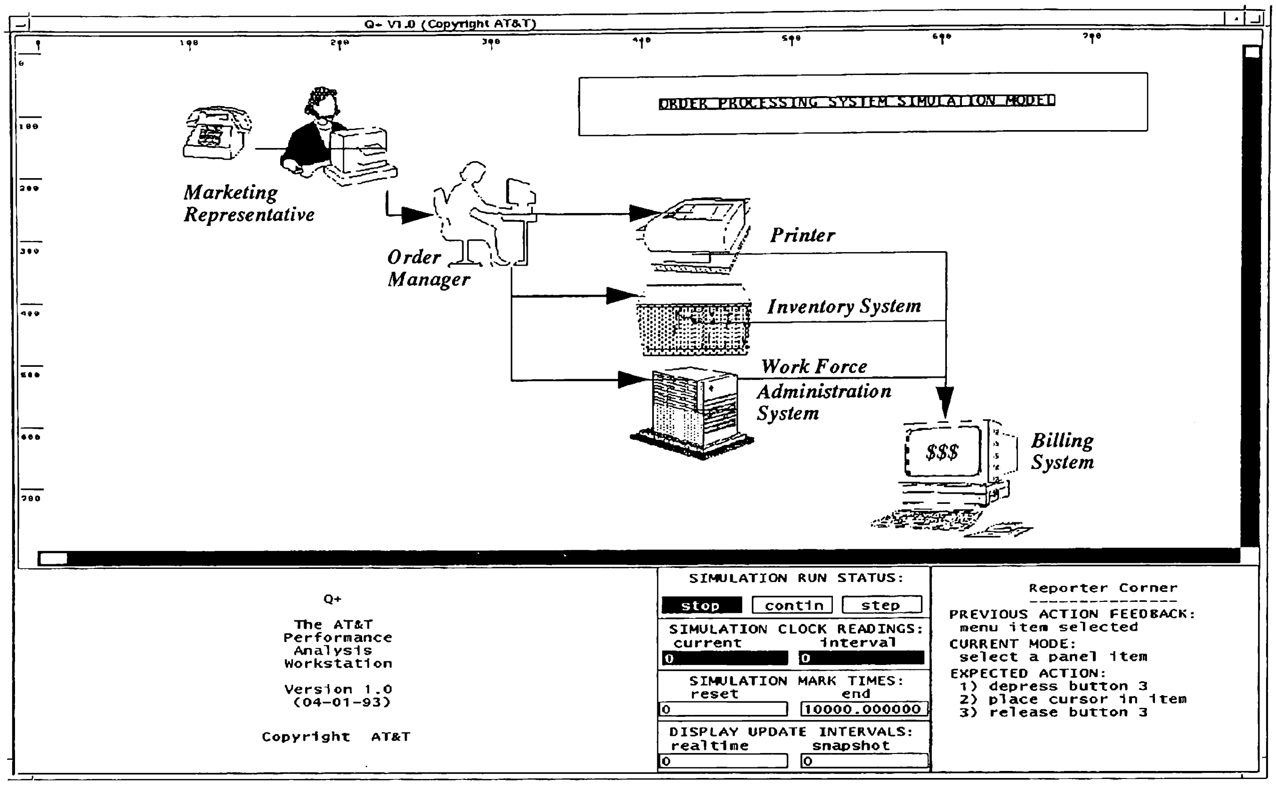
Both Figures 2 and 3 are Q+ models representing the above scenarios. Figure 2 shows the process view of the model. It is easily constructed by selecting appropriate icons that represent each step in the process. Each icon in fact is a queueing network model with its contents hidden. The details can be displayed, as shown in Figure 3, so actual performance characteristics, i.e., backlog of orders at a given step, utilization of resources, processing delays etc. can be observed during simulation. Note that the simulation can be executed with the contents of the subnetworks displayed or hidden.

In Figure 3, a service order, represented by the transaction SO flows through the queueing network from the Marketing queue to the ISSUE_ORDER node which splits the same SO transaction to the PRINT_ORDER node, the INVENTORY node and the WORK_FORCE node. After all three above queues have completed processing its respective SO transaction, a signal is sent to the BILLING node (which joins the SO signals from the above three nodes into a single SO transaction at the BILLING node) to start processing the bill for that order. The statistical window shows the histogram for total order processing time, that is, elapsed time since a marketing representative starts taking order to the time when a bill is ready to be sent to the customer (as tabulated by the BILLING node every time a SO transaction departs from the network).

4 The COMPASS MODELING INTERFACE

Historically, Q+'s user base mainly consisted of system and performance analysts. These users are very comfortable parametrizing simulation models using queueing theory terminology. However, one of the most powerful aspects of Q+ is that is general enough to model a wide range of applications, such as computer, communication, manufacturing, and process improvement systems, among others. As Q+'s user base has expanded to include increasing numbers of very targeted users with very specific modeling needs, we have realized that these users do not necessarily think of their systems in queueing theory terms. Instead of parameterizing simulation models in terms of nodes, classes, and transactions, they would much rather use terms that directly relate to their modeling scenario. For example, the process engineer designing the order processing system shown in Figure 2 would rather parameterize the model in terms of operator thinking time, customer interaction time, and so on, rather than edit all the parameters in each Q+ node directly.

The Complete Object Modeling interface for Performance Analysis SystemS (COMPASS) is designed for such purposes. We have developed interfaces that directly build Q+ models from application specific data. These interfaces are graphically equivalent, but differ in their input parameters and the output Q+ model. For example, the order processing system described previously, orders are routed over a Local Area Network (LAN) after a marketing representative keys them in. To determine the number of marketing representatives (clients) that the LAN and the order manager (server) can support, it will probably be necessary for a modeler to characterize the clients, the LAN and the Server in some detail. In Figure 4, we see the COMPASS windows for parameterizing a Client Group and the LAN. Here the user has entered a network speed of 10 MB/second, a packet size of 512 bytes, and background traffic at 100 packets/second. Also, the user has selected LAN utilization for observation. COMPASS will read the user inputs, then import the appropriate Q+ model (in this case, the Token Ring Network model), and parametrize the model correctly, e.g. a ratio of network

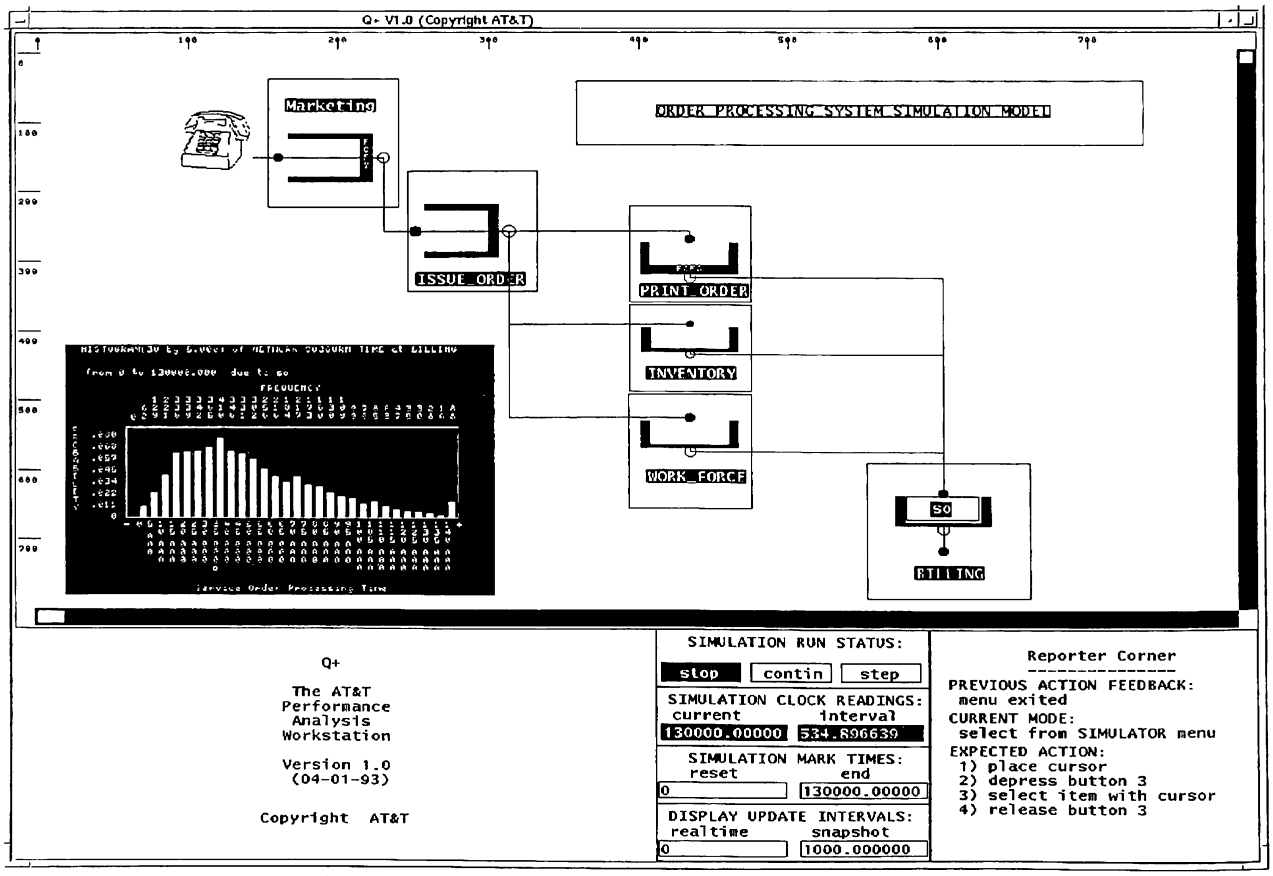


Q+
The AT&T Performance Analysis Workstation
Version 1.0 (04-01-93)
Copyright AT&T

SIMULATION RUN STATUS:
stop contin step
SIMULATION CLOCK READINGS:
current 0 interval 0
SIMULATION MARK TIMES:
reset 0 end 10000.000000
DISPLAY UPDATE INTERVALS:
realtime 0 snapshot 0

Reporter Corner
PREVIOUS ACTION FEEDBACK:
menu item selected
CURRENT MODE:
select a panel item
EXPECTED ACTION:
1) depress button 3
2) place cursor in item
3) release button 3

Figue 2. A Q+ Order Processing Model

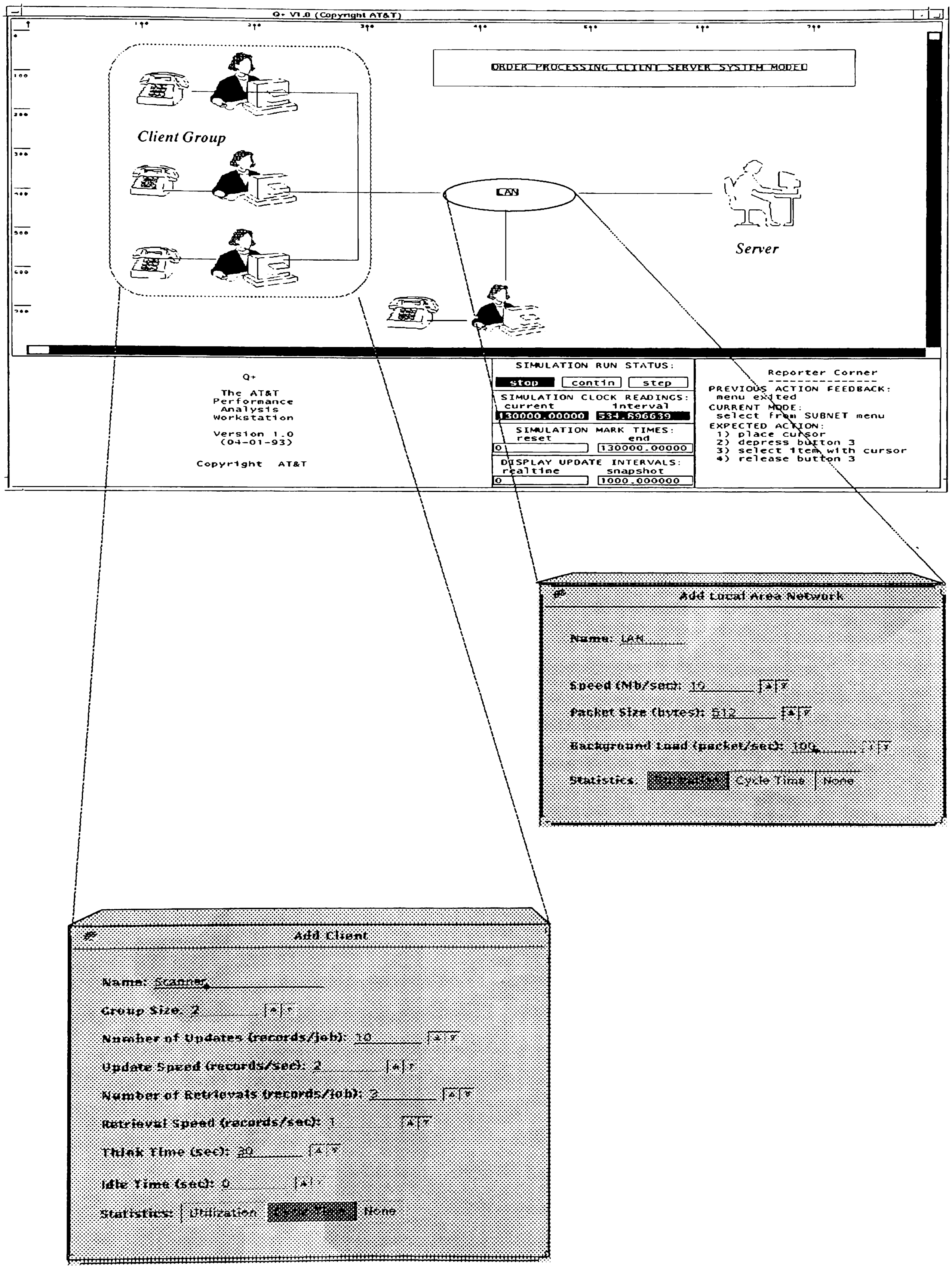


Q+
The AT&T Performance Analysis Workstation
Version 1.0 (04-01-93)
Copyright AT&T

SIMULATION RUN STATUS:
stop contin step
SIMULATION CLOCK READINGS:
current 130000.00000 interval 534.896639
SIMULATION MARK TIMES:
reset 0 end 130000.00000
DISPLAY UPDATE INTERVALS:
realtime 0 snapshot 1000.000000

Reporter Corner
PREVIOUS ACTION FEEDBACK:
menu exited
CURRENT MODE:
select from SIMULATOR menu
EXPECTED ACTION:
1) place cursor
2) depress button 3
3) select item with cursor
4) release button 3

Figure 3. Simulation Display of the Order Processing Model



Q+
The AT&T
Performance
Analysis
Workstation

Version 1.0
(04-01-93)

Copyright AT&T

SIMULATION RUN STATUS:

SIMULATION CLOCK READINGS:
current 130000.00000
interval 534.896639

SIMULATION MARK TIMES:
reset 0
end 130000.00000

DISPLAY UPDATE INTERVALS:
realtime 0
snapshot 1000.000000

```
Reporter Corner
---------------
PREVIOUS ACTION FEEDBACK:
 menu exited
CURRENT MODE:
 select from SUBNET menu
EXPECTED ACTION:
 1) place cursor
 2) depress button 3
 3) select item with cursor
 4) release button 3
```

Add Local Area Network

Name: LAN

Speed (Mb/sec): 10

Packet Size (bytes): 512

Background Load (packet/sec): 100

Statistics: [✓ [illegible]] Cycle Time None

Add Client

Name: Scanner

Group Size: 2

Number of Updates (records/job): 10

Update Speed (records/sec): 2

Number of Retrievals (records/job): 2

Retrieval Speed (records/sec): 1

Think Time (sec): 30

Idle Time (sec): 0

Statistics: Utilization [✓ [illegible]] None

Figure 4. The Q+ COMPASS Windows for the Order Processing Client Server Model

speed to packet size will determine the correct service time for the Message Transmission Queue of the Token Ring Network model and this will all be done transparently for the user.

Similarly, in Figure 4, we see a COMPASS window for parametrizing a Client Group. Here we are adding two marketing representatives to the client group named Scanners, with certain common update and retrieval parameters for the members of the group. The input to these parameters will be used to automatically generate a Q+ subnetwork model of a client group for use in performance analysis of order processing time.

COMPASS provides a wide range of default subnetwork models for LANs, X terminals, servers, host systems and networks. Once the user has selected the subnetworks and has completed parameterizing them, COMPASS builds the complete Q+ model from libraries of pre-existing models, in this case from the Computer Operations model library. COMPASS will also select appropriate Q+ statistics, run the model, then display the statistic results. Of course, since a Q+ model is built, the user can run the model on the Q+ screen instead. The benefit is that the user can run it interactively with animation, stopping the model and changing it at any time.

COMPASS is built using object oriented design principles and C++. This is a very natural paradigm since Q+ itself has a strong notion of objects, such as nodes, classes, transactions, etc. Anytime a new application arises, all that needs to be resolved is the mapping between the application objects and the corresponding Q+ objects. It is our goal to make this an automatic procedure so that end-users can create their own application interface.

While Q+ alone is quite powerful and flexible, using the COMPASS modeling interface will drastically simplify modeling building and interpretation.

5 CONCLUSION

In this paper, the authors have illustrated the use of Q+ to analyze the performance of a complex order processing system. The Q+ modeling software has proven to be a useful design tool for the design of computerized business operations systems. This paper has described the key design elements of Q+ and has demonstrated the power of its design approach.The last section on the COMPASS modeling interface described on-going work to enhance the use of Q+ for convenient modeling of computer operations systems.

REFERENCES

Baskett, F., K.M. Chandy, R.R. Muntz, and F.G. Palacios, "Open, Closed and Mixed Networks of Queues with Different Classes of Customers", ACM, Vol. 22, pp. 248-260, 1975.

Funka-Lea C. A., T.D. Kontogiorgos, R.J.T. Morris, L.D. Rubin, "Interactive Visual Modeling for Performance Analysis", IEEE Software, September, 1991.

Kelly F. P., Reversibility and Stochastic Networks, John Wiley, NY, 1979.

Melamed B. and R. J. T. Morris, "Visual Simulation: The Performance Analysis Workstation", IEEE Computer, Vol. 18, No. 8, Aug. 1985, 87-94.

Ramakrishnan K. J. and D. Mitra, "An Overview of PANACEA: A Software Package for Analyzing Markovian Queueing Networks", Bell System Technical Journal, Vol. 61, No. 10, pp. 2849-2815, 1982.

Whitt W., "The Queueing Network Analyzer", Bell System Technical Journal, Vol. 62, No. 9, pp. 2779-2815, 1983.

Q+ Version 1.0: User's Guide and Reference Manual (Volume 1), and Programmer's Guide and Reference Manual (Volume 2), Ed. by R. J. T. Morris. Available from C. L. Janczewski, AT&T Bell Laboratories, Room HO 3M328, Holmdel, NJ 07733, Tel. (908) 949-0678.

NOTICE OF DISCLAIMER

AUTHOR BIOGRAPHIES

Dennis S. Mok is a Member of Technical Staff at Bell Communications Research (Bellcore), Performance Management Department. He received his M. Eng. and Ph. D. in Industrial Engineering from Iowa State University. He was on the Computer Science faculty at Western Illinois University. His research interests are computer systems performance analysis, business processes and operations modeling and simulation, and complex process re-engineering. Dennis is a member of the editorial board of the International Journal in Computer Simulation.

Cynthia A. Funka-Lea is a Member of the Technical Staff at AT&T Bell Laboratories, Performance Analysis Department. Her research interests include simulation, object-oriented systems, and computer networks. Cynthia received a BS in computer science, BS in mathematics, and an MS in computer science from Penn State University. She is a member of the IEEE Computer Society and the ACM.